COMP3804 Midterm — 2h50m **Winter 2006**

Answer all questions in the exam booklet. In your answers you may use any algorithm discussed in class or on the assignments as a black-box. This exam contains 3 pages.

1 [3 marks] Recurrences

Solve the following recurrences by: (a) stating the number of nodes at level i in the recursion tree, (b) stating the size of each node at level i in the recursion tree, (c) stating the work done at each node at level i in the recursion tree, and (d) evaluating the sum of all the work done at all levels in the recursion tree.

1. $T(n) = 2T(n/3) + O(1)$
2. $T(n) = 4T(n/2) + O(n^2)$
3. $T(n) = 4T(n/3) + O(n)$

2 [1 mark] Fast Sorting(?)

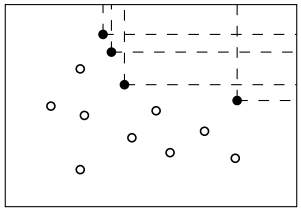
Name a sorting algorithm discussed in class that runs in $O(n^3)$ time.

3 [1 mark] Closest Pair

Let $A_1, \ldots, A_n$ be an array of real numbers. Describe an $O(n \log n)$ time algorithm to find the pair A_i and A_j, $i \neq j$ such that $|A_i - A_j|$ is minimum.

4 [1 mark] Maximal elements

Let $p_1, \ldots, p_n$ be a set of points in the plane, where $p_i = (x_i, y_i)$. We say that p_i is *maximal* if there is no p_j such that $x_j > x_i$ and $y_j > y_i$ (there is no point p_j in p_i's upper right quadrant; see the figure which shows the maximal points in black). Give an $O(n \log n)$ time algorithm to identify the maximal elements of $p_1, \ldots, p_n$.



5 [3 marks] Finding the Majority

Let $A_1, \ldots, A_n$ be an array of real numbers. Give an $O(n)$ time algorithm to determine if some element x occurs at least $\lceil n/2 \rceil$ times in the array.

Generalize this to give an $O(kn)$ time algorithm to determine some element x occurs $\lceil n/k \rceil$ times in the array.

6 [3 marks] Integer Element Uniqueness

Let $A_1, \ldots, A_n$ be an array of integers taken from the set $\{1, \ldots, m\}$. Give an $O(n + m)$ time algorithm to determine if there are two indices i and j, $i \neq j$ such that $A_i = A_j$.

For large values of m, the running time can be improved to $O(n(\log m / \log n))$. Explain how.

7 [1 mark] Incremental Location

Let $A_0, \ldots, A_n$ be an array of real numbers, with $A_0 = \infty$. Give an $O(n \log n)$ time algorithm to determine, for each i the smallest value in $A_0, \ldots, A_{i-1}$ that is greater than A_i (Find $k_1, \ldots, k_n$ where $k_i = \min\{A_j : 0 \leq j < i \text{ and } A_j > A_i\}$.

8 [1 mark] Boolean Matrix Multiplication

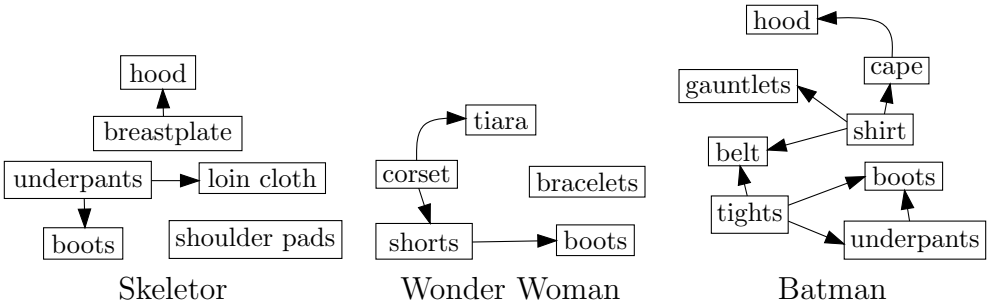
Let A and B be two $n \times n$ boolean matrices, with n a power of 2. The boolean matrix product $A \times B$ is the matrix C with

$$C_{ij} = (A_{i,1} \wedge B_{1,j}) \vee (A_{i,2} \wedge B_{2,j}) \vee \cdots \vee (A_{i,n} \wedge B_{n,j})$$

Describe how Strassen's algorithm could be used to compute the boolean matrix product in $O(n^{\log_2 7})$ time.

9 [2 marks] Super-Heroes and Super-Villains Get Dressed

Topologically sort the clothing worn by Skeletor, Wonder Woman and Batman.

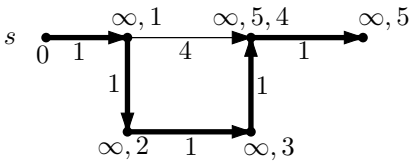


10 [1 mark+1 bonus] Shortest Uncommon Subsequence

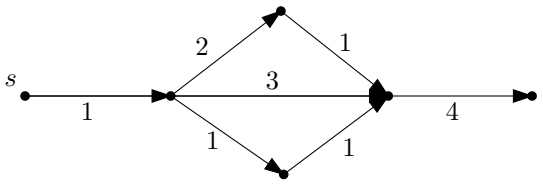
Design a fast algorithm that, given two sequences $s = \langle s_1, \ldots, s_n \rangle$ and $r = \langle r_1, \ldots, r_n \rangle$, computes the shortest subsequence of s that is not a subsequence of r. (Hint: To solve this, you really have to understand how the LCS problem is reduced to a graph problem. Save this one for last.)

11 [2 marks] Dijkstra's Algorithm

In the graph on the right below, illustrate the execution of Dijkstra's algorithm with source s by listing the values of $d(u)$ for each vertex u over time and by showing the resulting shortest-path tree. The figure on the left gives an example.



example



do this one

12 [3 marks] Fast Exponentiation

You are given a real number x and an integer n and you want to compute x^n using as few multiplications as possible. Show how, if n is a power of 2, you can compute x^n using $O(\log n)$ multiplications.

Show how to extend your algorithm to the case where n is not a power of 2. (Hint: consider the binary representation of n.)